View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Juelich Shared Electronic Resources

FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

Entwicklung einer maschinenunabhängigen Parallelversion eines nichtlinearen FE-Codes mit Kontaktalgorithmus (CONDAT-DYNA3D)

David Vinckier, Budi Saddak, Achim Basermann*

KFA-ZAM-IB-9524

September 1995
(Stand 30.10.95)

(*) CONDAT GmbH, D-85298 Fernhag

Erscheint in: Proceedings der HPSC 1995, Jülich, 11. - 14. September 1995

Verbundprojekt DYNA3D

Entwicklung einer maschinenunabhängigen Parallelversion eines nichtlinearen FE-Codes mit Kontaktalgorithmus (CONDAT-DYNA3D)

D. Vinckier, CONDAT GmbH, D-85298 Fernhag
B. Saddak, A. Basermann, KFA Jülich GmbH, ZAM, D-52425 Jülich

Kurzfassung

Von CONDAT-DYNA3D, einem Finite-Element-Programm aus dem Bereich nichtlinearer Strukturmechanik, wurde eine maschinenunabhängige Parallelversion entwickelt. Die Parallelisierung, die alle wichtigen Optionen, u.a. auch den Kontaktmodul, umfaßt, geschieht mittels Gebietszerlegung und Message-Passing-Techniken. Bei der Gebietszerlegung wurde das Multilevel Recursive Spectral Bisection Verfahren (MRSB) verwendet, das in kurzer Zeit eine fast optimale Zerlegung erreicht. Rechenbeispiele zeigen bei typischen industriellen Problemen mit ca. 60000 Schalen- oder Volumenelementen einen guten Beschleunigungswert von z.B. 44 bei 50 Prozessoren auf einer Intel-Paragon oder 1.9 auf einem Workstation-Verbund mit zwei HP 735 Rechnern und einer TCP/IP Ethernetverbindung. Eine analytische Abschätzung des Speedup-Verhaltens ermöglicht eine realistische Bewertung der erzielten Ergebnisse und die Vorhersage für andere Hardware-Plattformen.

1 Einleitung

Simulationsprogramme mit expliziter Zeitintegration aus dem Bereich nichtlinearer Strukturmechanik gewinnen zunehmend an Bedeutung. Der Anwendungsbereich dieser Programme ist sehr breit und reicht von der Simulation quasistatischer Materialprüfungstests, der Simulation von Crash-Tests über die Berechnung von impaktresistenten Schutzstrukturen bis hin zur Simulation von Hochgeschwindigkeitsvorgängen. CONDAT wendet sein Programm CONDAT-DYNA3D zur Durchführung von Projekten im Auftrag deutscher Industrieunternehmen in allen diesen Bereichen an (siehe z.B. [2], [3]). Bedingt durch die hohe Elementzahl (typischerweise 50000 bis 600000 Freiheitsgrade) und den aus Stabilitätsgründen sehr kleinen Zeitschritten (typischerweise 0.01 bis 5 μs !) ist der Rechenaufwand dieser Klasse von Programmen gerade bei industriellen Beispielen sehr hoch. Um zu klären, ob eine effiziente, allgemeine – d.h. maschinenunabhängige – Parallelisierung derartiger Programme möglich ist, die auch zu einer angemessenen Leistungssteigerung führt, wird in einem vom BMBF geförderten Projekt eine maschinenunabhängige Parallelversion von CONDAT-DYNA3D entwickelt. Die erreichte Rechenleistung sowie das Entwicklungspotential auf unterschiedlichen Parallelrechnern wird ermittelt und kritisch beurteilt.

2 Parallelisierungskonzept

Explizite Finite-Element-Verfahren verwenden eine direkte Zeitintegration der Bewegungsgleichungen zur Lösung des Gleichungssystems. Dies erfordert bei einer Parallelisierung mindestens eine Synchronisation pro Zeitschritt. Damit stehen für die Parallelisierung jedes einzelnen Zeitschrittes grundsätzlich zwei Konzepte zur Auswahl:

- *Mikro-Parallelisierung*: Jeder einzelne Berechnungsabschnitt (z.B. für ein FORTRAN Programm jede `SUBROUTINE`) wird parallelisiert.
- *Makro-Parallelisierung*: Der ganze Berechnungsablauf für einen Zeitschritt wird parallel ausgeführt. Dies kann über eine Gebietszerlegung und einen Informationsaustausch für die Gebietsränder realisiert werden. Dabei besteht der Gebietsrand nur aus Knoten, weil die Gleichungen explizit integriert werden.

Die Makro-Parallelisierung hat folgende Vorteile:

- Der Programmieraufwand ist beschränkt, weil neben der notwendigen Synchronisation des Zeitschrittes nur Knotenkräfte an den Gebietsrändern ausgetauscht werden müssen, wie in Abbildung 1 dargestellt ist. Die eigentliche Berechnung dieser Kräfte bleibt im Vergleich zum sequentiellen Programm unverändert. Damit sind auch zukünftige Programmoptionen (neue Elementtypen, neue Materialmodelle) automatisch parallelisiert. Lediglich im Kontaktmodul muß der sequentielle Algorithmus erweitert werden.
- Der Informationsaustausch geschieht durch *Message Passing*. Diese Technik ist auf den meisten Parallelrechnern mit verteiltem Speicher implementiert (PVM, MPI, ...). Die wenigen maschinenabhängigen Aufrufe können als getrennte Module realisiert werden, so daß das FE-Programm maschinenunabhängig bleibt.
- Das Programm kann ohne weiteres in einem heterogenen Rechnersystem (z.B. Workstation-Verbund) eingesetzt werden.
- Die Gebietszerlegung kann in einem vom Lösungssystem unabhängigen Modul realisiert und getestet werden.

Auf einer Shared-Memory-Architektur ist der Informationsaustausch durch Message Passing in der Regel nicht die günstigste aber eine mögliche Methode (PVM z.B. ist häufig vorhanden). Auf derartigen Systemen sind gewöhnlich effizientere und einfachere Programmiermodelle vorhanden. Weiter kann bei Modellen mit intensiven und wechselnden Kontakten die optimale Gebietszerlegung in jedem Berechnungsabschnitt unterschiedlich sein. In diesem Fall führt eine Rechnung mit unveränderter Zerlegung zu keinem optimalen Lastausgleich.

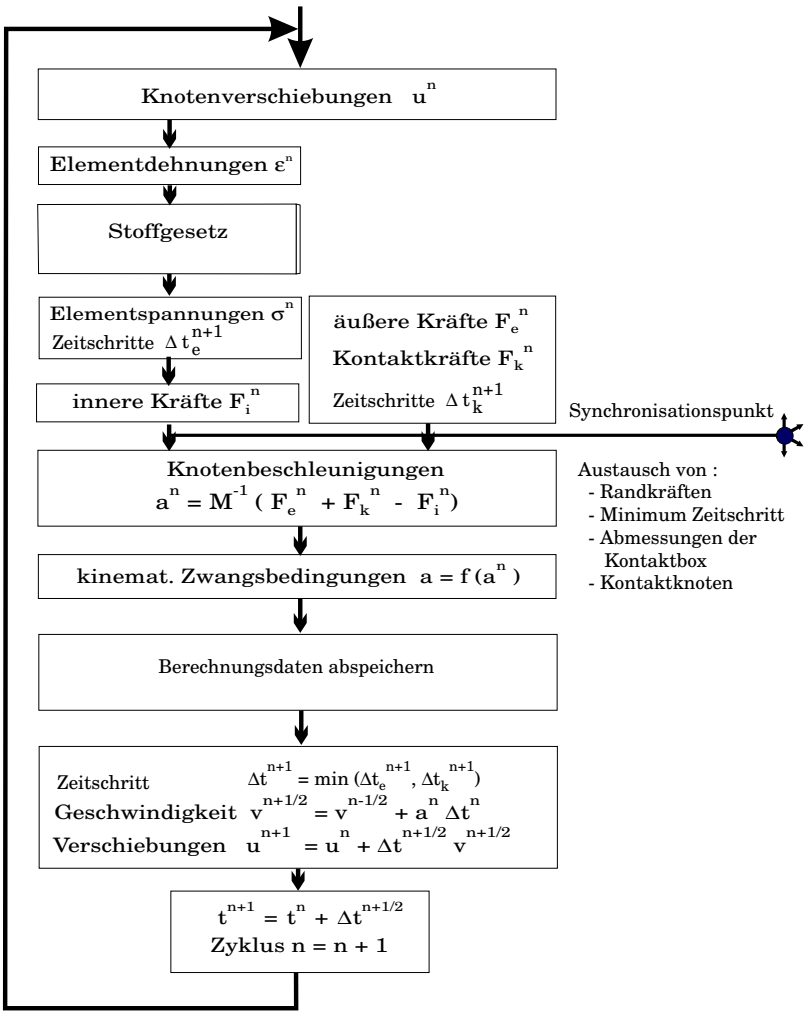

Abbildung 1: Rechenzyklus in dem parallelisierten CONDAT-DYNA3D mit Synchronisationspunkt

3 Gebietszerlegung nach der MRSB-Methode

Das Ziel eines jeden Gebietszerlegers ist, eine optimale Partitionierung zu erhalten. Auf einem Parallelrechner mit verteiltem Speicher heißt dies im wesentlichen, eine gleichmäßige Lastverteilung und eine minimale Kommunikation zu erzielen. Die Probleme sind meistens komplex und groß. Es ist deshalb nicht einfach, ein Verfahren zu entwickeln, welches sowohl eine optimale Partitionierung liefert als

auch schnell genug die Ergebnisse berechnet. Das hier verwendete Verfahren *Multilevel Recursive Spectral Bisection* (MRSB) [1] basiert auf einem graphentheoretischen Ansatz und ermöglicht eine effiziente Lösung des sich dabei ergebenden Eigenwertproblems.

3.1 Graphentheoretischer Ansatz

Da die Gebietszerlegung für beliebige Diskretisierungsgitter verwendet werden soll, muß gewährleistet werden, daß die Zerlegung auf keine bestimmte Geometrie zugeschnitten ist. Um dies zu erreichen, wird die Problemlösung auf die graphentheoretische Ebene übertragen. Das Diskretisierungsgitter wird durch einen ungerichteten, zusammenhängenden Graphen aus N Knoten und M Kanten repräsentiert. Dabei können die Knoten finite Elemente oder auch Gitterpunkte eines Körpers sein. Kanten verbinden einen Knoten mit den anderen Knoten, die bei der diskreten Lösung zur Berechnung der Werte an diesem Punkt oder Element herangezogen werden. In der Regel sind dies angrenzende Punkte oder Elemente. Abbildung 2 zeigt das Beispiel eines Graphen G mit 12 Knoten und 13 Kanten.

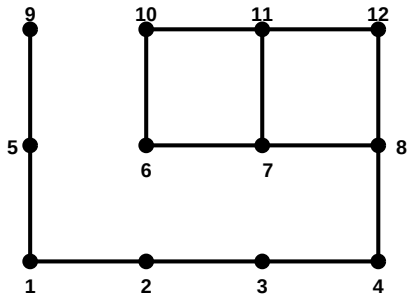

Abbildung 2: Graph G

Für eine optimale Zerlegung muß das Problem in dieser Ebene anders formuliert werden. Eine gleichmäßige Lastverteilung wird erreicht, indem man jedem Gebiet die gleiche Anzahl von Knoten zuteilt. Wenn alle Knoten eines Graphen G auf zwei nicht zusammenhängende Mengen A, B verteilt werden und die Anzahl der Knoten in A, B gleich ist, liegt *Bisektion* vor. Das Problem des minimalen Datenaustauschs wird mit Hilfe der geschnittenen Kanten gelöst. Je weniger Kanten man schneidet, umso weniger Kommunikation liegt vor. Auf dieser Strategie basierend haben Pothen, Simon und Liou das RSB-Verfahren entwickelt [4]. Im folgenden wird das Verfahren an Hand des Beispiels aus Abbildung 2 erläutert.

3.2 Laplace-Matrix und Fiedler-Vektor

Zunächst wird für das Verfahren die Laplace-Matrix L des Graphen $G = (V, E)$ bestimmt. V ist die Menge der Knoten und E die Menge der Kanten. Da der Graph zusammenhängend ist, ist die Matrix *positiv semidefinit*. Sie ist ähnlich der Adjazenzmatrix aufgebaut; sind zwei Knoten im Graphen durch eine gemeinsame Kante verbunden, wird in die Matrix die Zahl –1 eingetragen, sonst eine Null. In die Diagonale wird die Zahl der von einem Knoten abgehenden Kanten eingetragen. Diese Zahl wird als Grad (degree) eines Knotens bezeichnet. Daher ist die Zeilensumme der Laplace-Matrix stets Null. Das Bildungsgesetz lautet somit:

$$l_{ij} = \begin{cases} -1 & \text{falls } (v_i, v_j) \in E \\ \text{degree}(v_i) & \text{falls } i = j \text{ und } v_i \in V \\ 0 & \text{sonst} \end{cases} \tag{1}$$

Auf unser Beispiel bezogen lautet die Laplace-Matrix:

$$L = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 3 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 2 \end{pmatrix} \tag{2}$$

Der kleinste Eigenwert der positiv semidefiniten Laplace-Matrix ist stets Null. Von dieser Matrix wird der zweitkleinste Eigenwert bestimmt, z.B. mit Hilfe des *Lanczos-Algorithmus*; anschließend wird der dazugehörige Eigenvektor, der *Fiedler-Vektor* berechnet. Den Komponenten dieses Vektors werden die Knotennummern des Graphen zugeordnet. Für die Matrix L erhält man als zweitkleinsten Eigenwert $\lambda = 0.0913$. Im linken Teil von (3) sind die Komponenten des Fiedler-Vektors und in eckigen Klammern die zugeordneten Knotennummern des Graphen aufgeführt. Rechts in (3) sind die Komponenten des Fiedler-Vektors der Größe nach sortiert. Auf die zugehörigen Knotennummern wird die gleiche

Permutation angewandt.

$$\begin{pmatrix} -0.348 & [1] \\ -0.234 & [2] \\ -0.098 & [3] \\ 0.047 & [4] \\ -0.431 & [5] \\ 0.288 & [6] \\ 0.257 & [7] \\ 0.187 & [8] \\ -0.474 & [9] \\ 0.294 & [10] \\ 0.272 & [11] \\ 0.240 & [12] \end{pmatrix} \longrightarrow \begin{pmatrix} -0.474 & [9] \\ -0.431 & [5] \\ -0.348 & [1] \\ -0.234 & [2] \\ -0.098 & [3] \\ 0.047 & [4] \\ 0.187 & [8] \\ 0.240 & [12] \\ 0.257 & [7] \\ 0.272 & [11] \\ 0.288 & [6] \\ 0.294 & [10] \end{pmatrix} \tag{3}$$

3.3 RSB-Methode

Nachdem man den Fiedler-Vektor bestimmt und die Reihenfolge der Knotennummern erhalten hat, kann die Verteilung der Gitterknoten erfolgen. Bei einer Aufteilung des Graphen aus Abbildung 2 auf zwei Gebiete werden je sechs Knoten auf diese verteilt. Die Knoten mit den ersten sechs Nummern des permutierten Numerierungsvektors ganz rechts in (3) werden dem einen Gebiet zugeteilt, die letzten sechs dem anderen. Durch diese Zuordnung erhält man die Zerlegung nach Abbildung 3. Man erkennt, daß eine optimale Zerlegung erzielt wird, da jedes Gebiet sechs Punkte enthält und lediglich eine Kante geschnitten wird. auf einem Parallelrechner mit verteiltem Speicher bedeutet dies gleichmäßige Lastverteilung und minimale Kommunikation.

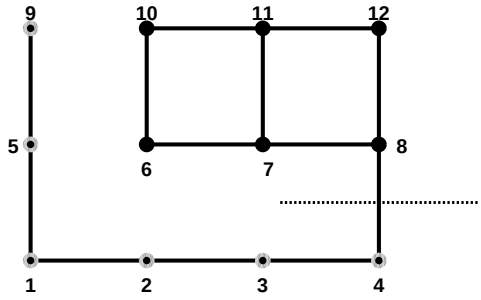

Abbildung 3: Gebietszerlegung des Graphen G

3.4 Multilevel-Methode

Die meiste Rechenzeit beim RSB-Verfahren erfordert die Berechnung des Fiedler-Vektors. Je größer die Anzahl der Knoten des Graphen ist, umso länger dauert die Berechnungszeit. Um den Fiedler-Vektor effizienter berechnen zu können, wurde die sogenannte *Multilevel*-Methode entwickelt [1]. Sie besteht im wesentlichen aus vier Schritten. Im ersten Schritt führt man eine *Kontraktion* des Graphen durch, d.h. man teilt den ursprünglichen Graphen, dessen Fiedler-Vektor man berechnen will, in mehrere Untergraphen. Von diesen Untergraphen berechnet man im zweiten Schritt den Fiedler-Vektor, z.B. mit Hilfe des Lanczos-Algorithmus. Im dritten Schritt aproximiert man den Fiedler-Vektor des ursprünglichen Graphen durch Interpolationsmethoden. Als letzter Schritt wird eine Verbesserung des Fiedler-Vektors durch *Rayleigh-Quotienten*-Iteration durchgeführt. Der letzte Schritt ist nicht zwingend notwendig. Meist kann die Iteration nach zwei Schritten abgebrochen werden, da gewöhnlich die Verbesserung bereits ausreichend ist. Somit bedeutet diese Iteration kaum einen Mehraufwand an Rechenzeit.

4 Parallelisierung durch Gebietszerlegung

In den hier betrachteten Anwendungsproblemen wird die Gebietszerlegung ein einziges Mal zu Beginn durchgeführt. Alle übrigen Berechnungen finden parallel auf den einzelnen Prozessoren statt. Die Zerlegung wird sequentiell durchgeführt, da Zeitmessungen ergaben, daß im Vergleich zur Simulationszeit der Aufwand für die Zerlegung gering ist. Der Zerleger liest zunächst ein bestimmtes Inputfile ein. Mit Hilfe des implementierten MRSB-Verfahrens werden die Daten des Gebietes zerlegt. Anschließend werden die Inputfiles generiert, die die Schnittstelle vom Gebietszerleger zum Simulationsprogramm bilden.

5 Ein paralleler Kontaktalgorithmus

Charakteristisch für das Kontaktmodul im Vergleich zum *Finite-Element-Modul* ist, daß die Anzahl der Kontakte ständig wechseln kann und die große Menge aller a priori möglichen Kontakte eine Vorsortierung erfordert, um den Berechnungsaufwand zu minimieren. Dazu wird um jedes Kontaktsegment eine Box konstruiert und danach die Liste der Knoten in dieser Box ermittelt. Nur für diese Knoten/Segment-Paare werden die Kontaktberechnungen durchgeführt. Die Liste mit diesen „Kontaktkandidaten“ wird in vom Programm bestimmten Intervallen ständig aktualisiert. Dieses Verfahren ist in [5] beschrieben. Für die Parallelisierung wurde zunächst nur die Kontaktoption mit Rückstellfedern in Betracht genommen, weil in dem Fall alle Kontaktkräfte unabhängig voneinander berechnet werden können. Bei einem zerlegten Modell kann man zwischen lokalen und externen Kontakten unterscheiden (siehe Abbildung 4). Die lokalen

Kontakte entstehen zwischen Knoten und Segmenten innerhalb eines Gebietes und können wie im sequentiellen Fall behandelt werden. Externe Kontakte entstehen zwischen Knoten und Segmenten, die verschiedenen Gebieten angehören. Externe Kontakte werden ebenfalls wie lokale Kontakte behandelt, indem man den Gebietsrand um die Kontaktknoten erweitert. Dazu wird auch hier eine Box konstruiert, diesmal um das ganze Gebiet, und alle Kontaktknoten der anderen Gebiete, die sich in dieser Box befinden, werden dem Gebietsrand zugeordnet. Da jedes Segment nur einmal vorkommt und immer in seinem ursprünglichen Gebiet bleibt, ist sichergestellt, daß jeder Kontakt nur einmal behandelt wird.

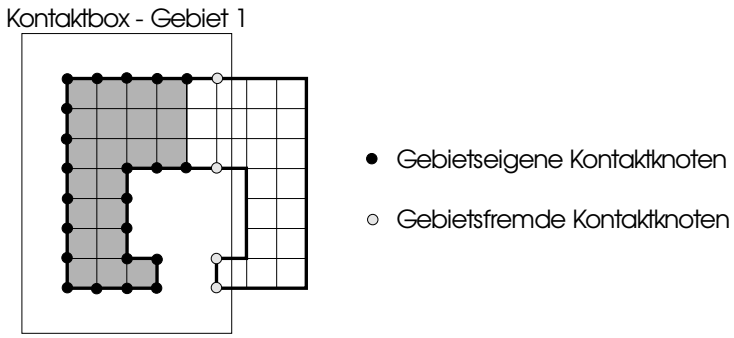

Abbildung 4: FE-Modell mit Kontaktknoten für Gebiet 1

Diese Methode hat den Vorteil, daß kaum Änderungen im Kontaktmodul notwendig sind und das Programm mit einer Gebietszerlegung am Anfang der Berechnung auskommt. Der Nachteil ist, daß bei wechselnden Kontakten leicht eine ungünstige Lastverteilung entstehen kann, die nur durch eine neue Gebietszerlegung zu beheben ist. Damit der Gebietsrand nicht zu viele Knoten hat, muß bei der Gebietszerlegung auch die Kastenform der Gebietsbox berücksichtigt werden. Dies bedeutet, daß eine rein algebraische Zerlegung hier nicht unbedingt optimal ist.

6 Skalierbarkeit des parallelisierten Programms

Die Skalierbarkeit ist ein Maß dafür, wie effektiv ein Programm bei steigender Prozessorzahl die vorhandene Rechnerleistung ausnutzen kann. Der Idealfall der linearen Skalierbarkeit ist bei der Parallelisierung durch Gebietszerlegung zum einen wegen der erforderlichen Kommunikation zwischen den Prozessoren nicht erreichbar, zum anderen, weil der parallele Algorithmus im Vergleich zum sequentiellen Fall einen gewissen, wenn auch kleinen Mehraufwand erfordert. Um die

Skalierbarkeit des Lösers auf einer bestimmten Architektur analytisch zu bestimmen, muß man die mittlere Rechenzeit pro Einzelzeitschritt in Betracht ziehen. Obwohl diese analytische Abschätzung nicht unproblematisch ist, wird hier versucht, die wichtigsten Faktoren zu erfassen. Bei Modellen ohne Kontakt ist diese Rechenzeit T_c quasi konstant und für jeden Elementtyp und jedes Materialmodell proportional zur Anzahl der Elemente N_e. Bei Modellen mit Kontakt variiert der zusätzliche Rechenaufwand je nach Anzahl der aktiven Kontaktsegmente N_s, der sortierten Kontaktknoten per Segment N_{ns}, der aktiven Kontakte N_{ac} und je nach Frequenz f_{cs}, mit der die Liste der Kontaktpaare aktualisiert werden muß. Damit kann die Rechenzeit pro Zeitschritt für den sequentiellen Fall bei einem Modell mit einem Elementtyp und Kontakt annähernd wie folgt beschrieben werden:

$$T_c = T_v + T_{ec} \cdot N_e + T_{scc} \cdot N_s \cdot N_{ns} \cdot f_{cs} + T_{fcc} \cdot N_{ac} \tag{4}$$

mit T_{ec}, T_{scc}, und T_{fcc} den spezifischen Rechenzeiten pro Element, pro Sortierpaar und pro Kontakt, die jeweils vom Algorithmus und vom Rechnertyp abhängig sind, und T_v eine konstante Verwaltungszeit. Für den parallelen Fall wird die Rechenzeit pro Zeitschritt T^i_{cp} für den Prozessor i berechnet als:

$$T^i_{cp} = T_v + T_{ec} \cdot N^i_e + T_{scc} \cdot N^i_s \cdot N^i_{ns} \cdot f_{cs} + T_{fcc} \cdot N^i_{ac} + T^i_0 + T^i_m \ . \tag{5}$$

Der Mehraufwand des parallelen Algorithmus T^i_0 kann genauso wie z.B. T_{ec} aus Versuchen abgeleitet werden. Die Kommunikationszeit T^i_m berechnet sich aus der Länge und Anzahl der Datentransfers und aus der maschinenspezifischen Latenzzeit und Bandbreite. Der Beschleunigungswert S kann jetzt abgeleitet werden aus:

$$S = \frac{T_c}{\max\limits_i(T^i_{cp})} \ . \tag{6}$$

Bei Berechnungen mit Kontakt ist die praktische Auswertung von (6) mittels (4) und (5) nicht immer möglich, weil die kontaktspezifischen Größen geometrisch bedingt sind und je nach Zerlegung stark unterschiedlich sein können. Für ein Modell ohne Kontakt kann man den Beschleunigungswert für p Prozessoren wie folgt abschätzen:

$$S = p \cdot \frac{T_c}{\frac{T_c - T_v}{Q_z} + (T_v + \max\limits_i(T^i_m) + \max\limits_i(T^i_0))} \tag{7}$$

mit $Q_z = \dfrac{N_e}{p \cdot \max\limits_i(N^i_e)}$. Für die Simulation einer Wabeneindrückung (Benchmark P7), die hier zunächst ohne Kontakt betrachtet wird, ist dieser Beschleunigungswert für zwei Rechnerarchitekturen in Abbildung 5 dargestellt. Es muß dabei erwähnt werden, daß in dieser Abschätzung bestimmte Einflüsse (z.B. Verwaltungsaufwand des Programms, Optimierungsmöglichkeiten des Compilers, ...) nicht oder nur ungenau erfaßt werden und daher der hier berechnete Beschleunigungswert nur eine erste Näherung darstellt.

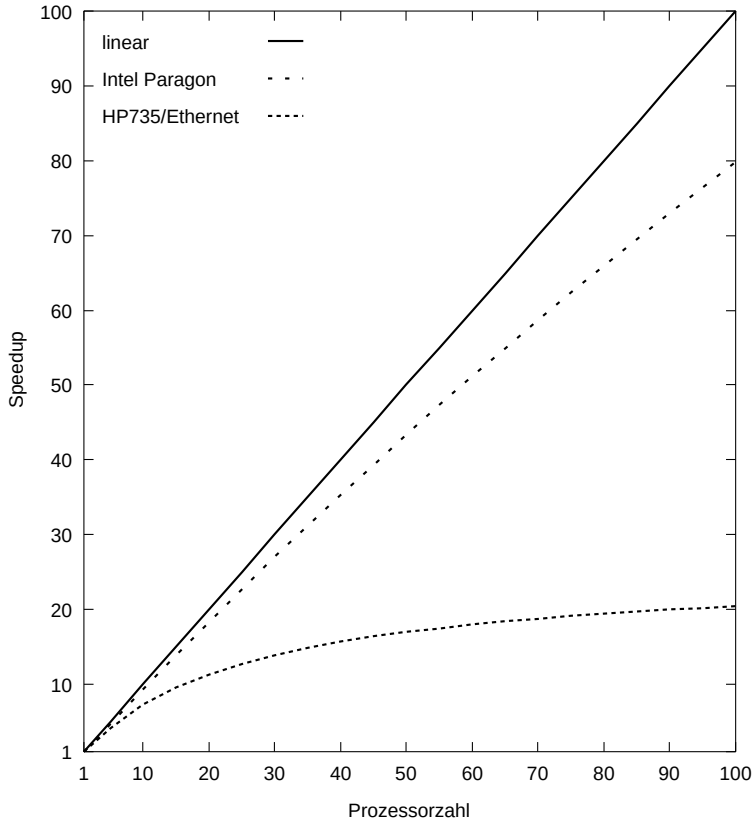

Abbildung 5: Berechneter Speedup für die Wabeneindrückung

7 Ergebnisse

Es wurden zunächst drei Beispiele auf der Paragon XP/S 10 der KFA Jülich gerechnet. Das erste ist ein Stab, der auf eine feste Wand aufprallt, bestehend aus 12573 Knoten und 9120 Vollelementen. Das zweite Rechenbeispiel ist ein S-Träger, der auf eine Wand gedrückt wird. Er besteht aus 6178 Knoten und 6168 Schalenelementen. Tests ergaben bei 16 Rechenknoten eine parallele Effizienz über 50 Prozent. Der Einsatz höherer Prozessorzahlen ist bei der geringen Problemgröße nicht sinnvoll. Das dritte Beispiel ist ein Körper mit einer Honigwabenstruktur wie in Abbildung 6 dargestellt. Es besteht aus 56862 Knoten und 59648 Schalenelementen. Dieses Beispiel wurde zunächst auf bis zu 50 Rechenknoten gerechnet. Bei diesem deutlich größeren Beispiel konnten nahezu optimale Speedups erzielt werden, wie Abbildung 7 zeigt. Auf einem Workstation-Cluster der Firma CONDAT bestehend aus zwei HP 735 Rechnern wurde für dasselbe Beispiel ein Speedup von 1.9 erzielt.

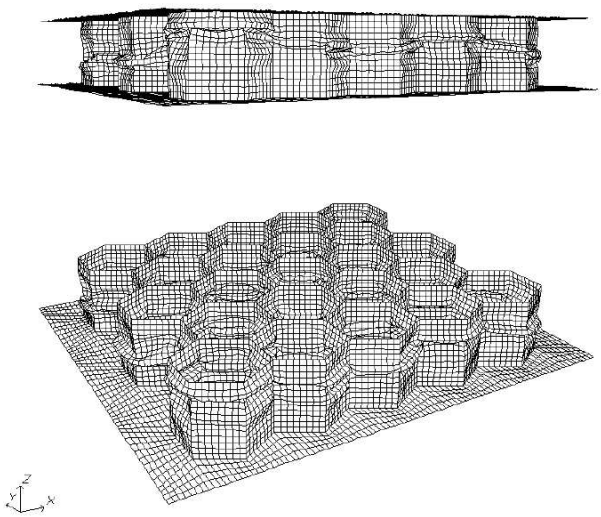

Abbildung 6: Wabeneindrückung

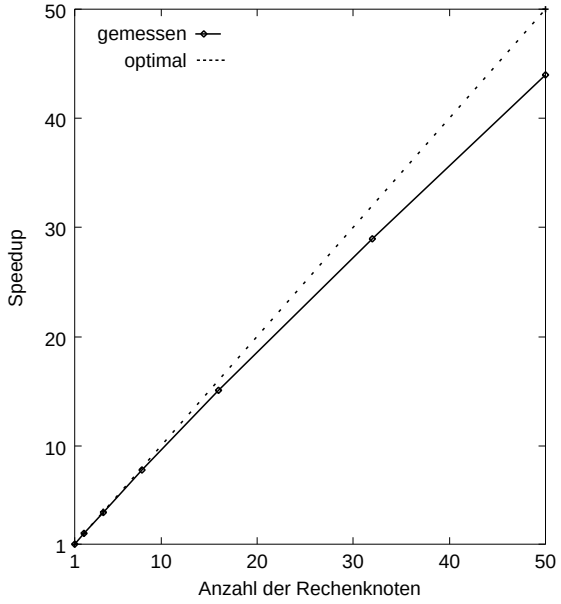

Abbildung 7: Speedup-Messung

8 Schlußfolgerungen

Die Parallelisierung eines Finite-Element-Verfahrens mit expliziter Zeitintegration aus dem Bereich nichtlinearer Strukturmechanik (CONDAT-DYNA3D) hat gezeigt, daß mit der Methode der Gebietszerlegung, kombiniert mit Message-Passing-Techniken, ohne größeren Programmieraufwand in kurzer Zeit eine gute Rechenleistung erreicht werden kann. Dies gilt sowohl für echte Parallelrechner (z.B. Intel Paragon) als auch für Workstation-Cluster.
Speziell bei Modellen mit Kontaktvorgängen (mit Abstand die wichtigste Kategorie bei industriellen Anwendungen) sind noch Verbesserungen bezüglich einer dynamischen Zerlegung möglich. Eine bessere Abstimmung zwischen Zerlegermodul und Kontaktmodul des Finite-Element-Verfahren, sowie die Realisierung einer dynamischen Zerlegung versprechen eine weitere Verbesserung der Leistung und der Flexibilität des entwickelten Programms auf parallelen Hochleistungsrechnern.

Literatur

[1] S. T. Barnard and H. D. Simon, *Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems*, Concurrency: Practice and Experience, 6(2) (1994), pp. 101–117.

[2] M. Maier, V. Altstädt, D. Vinckier, and K. Thoma, *Härtetest für Faserverbundwerkstoffe im Computer*, Werkstoff und Innovation, 1991.

[3] M. Maier, V. Altstädt, D. Vinckier, and K. Thoma, *Numerical and experimental analysis of the compact tension test for a group of modified epoxy resins*, Journal of Polymer Testing, 13 (1994), pp. 55–66.

[4] A. Pothen, H. D. Simon, and K. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Mat. Anal. Appl., 11 (1990), pp. 430–452.

[5] K. Thoma and D. Vinckier, *Numerical analysis of a high velocity impact of fiber reinforced materials*, IMPACT IV, Post-Conference Seminar of the 12th Int. Conference on Structural Mechanics in Reactor Technology (SMiRT), 1993.